Heterogeneous Timed Machines

Benoit Delahaye, J. L. Fiadeiro, Axel Legay, Antónia Lopes

Heterogeneous Timed Machines

Benoît Delahaye[1], José Fiadeiro[2], Axel Legay[2,3], and Antónia Lopes[4]

[1]Université de Nantes / LINA, France
benoit.delahaye@univ-nantes.fr
[2]Dep. of Computer Science, Royal Holloway University of London, UK
jose.fiadeiro@rhul.ac.uk
[3]INRIA/IRISA, Rennes, France
axel.legay@irisa.fr
[4]Dep. of Informatics, Faculty of Sciences, University of Lisbon, Portugal
mal@di.fc.ul.pt

Abstract. We present an algebra of discrete timed input/output automata that execute in the context of different clock granularities — timed machines — as models of systems that can be dynamically interconnected at run time in a heterogeneous context. We show how timed machines can be refined to a lower granularity of time and how timed machines with different clock granularities can be composed. We propose techniques for checking whether timed machines are consistent or feasible. Finally, we investigate how consistency and feasibility of composition can be proved at run-time without computing products of automata.

1 Introduction

Many software applications operating in cyberspace need to connect, dynamically, to other software systems. For example, in the domain of intelligent transportation, systems for congestion avoidance or coordination of self-driven convoys of cars need to be able to accommodate interconnections that are established at run time between components that cannot be pre-determined at design time.

Applications such as these often have real-time requirements, i.e., their correctness depends not only on what outputs are returned to given inputs, but also on the time at which inputs are received and corresponding outputs are produced and communicated. When components of such software applications, usually written in a high-level programming language and relying on particular time abstractions, are executed in a given execution platform, their real-time behaviour is additionally restricted by the clock period of that platform. Components interconnected at run time will be likely to operate over different clock periods, resulting in a timed heterogeneous system.

Existing formalisms for modeling time-constrained systems focus mainly on mono-periodic systems, i.e., they assume that all system components will operate over a shared clock period. These models can still be used for timed heterogeneous systems whose structure is fixed and known *a-priori* by modeling the system components in terms of a global clock that is the least common multiple

of all local clocks. In the case of systems whose structure is dynamic and defined at run time, this is no longer possible [3].

In this paper, we propose a formal model for timed heterogeneous systems that does not require *a-priori* knowledge of their composition structure. Our model is based on input/output automata and supports *run-time compositionality* in the following sense: it is possible to ensure that components can work together as interconnected over heterogeneous local clocks by relying only on properties of models of those components, with no need for calculating their composition. More specifically, we provide the means to determine if the interconnection of two automata is consistent (there is at least a joint execution) or feasible (there is at least a joint execution no matter what inputs the components receive from their environment) not by calculating and checking properties of their product at run time but by relying on properties of the individual automata that can be established at design or composition time. Those properties ensure that the automata are able to co-operate at run time without modifying their time domains.

Our starting point is the homogeneous timed component algebra that we proposed in [8] for services. The extension from a homogenous to a heterogeneous setting is not trivial (which justifies this paper) because, where the algebraic properties of composition in a homogenous-time domain generalise those of the un-timed domain, interconnection in a heterogeneous setting is not even always admissible. For that reason, the algebra that we propose in Sec. 3 separates the space of discrete timed input/output automata (TIOA) [14, 7] from that of their executions over a given clock: the components of our algebra are pairs of a TIOA and a clock granularity, what we call timed machines. Two operations are defined over timed machines: *heterogeneous composition*, which extends the traditional product of TIOA to the situation in which the granularities of the two machines are not the same, and *refinement*, which extends a machine with new states and transitions in order to accommodate a finer clock granularity as required to interoperate with other machines. Still, refinement does not reduce heterogeneous composition to the homogeneous setting, which leads us to define a notion of 'best approximation' through which we can characterise classes of timed machines that can be used to reason about or simulate interconnections of timed machines with commensurable clock granularities.

In Sec. 4, we study two important properties when modelling systems: *consistency*, in the sense that a machine can be ensured to generate a non-empty language, and *feasibility*, in the sense that a machine can be ensured to generate a non-empty language no matter what inputs it receives. Finally, we prove two compositionality results, one for consistency and the other for feasibility. Those results rely on a number of properties that can be checked, at design time, over given timed machines to ensure that their interconnection will be consistent or feasible without actually having to calculate the product of the corresponding automata at run time. Those properties ensure that components that implement the timed machines can work together across different clock granularities.

2 Preliminaries

2.1 Timed traces

Although transition systems are typically used as operational semantics of automata (including timed transitions systems for timed automata as in [13]), we use instead a trace semantics because the topological properties of trace domains allow us to provide a finer characterisation of properties such as consistency and feasibility (cf. Sec. 4). For example, existing transition-system semantics such as [7] offer a weaker notion of consistency for timed automata because it fails to enforce time progression and, therefore, an automaton that does not accept any non-Zeno timed sequence can still be consistent. The proposed operational semantics is also much closer to the one that we used in the homogeneous-timed [8] and un-timed [9] domains, thus making it easier to understand the challenges raised by a heterogeneous domain.

We start by recalling a few concepts related to traces. Given a set A, a *trace* λ over A is an element of A^ω, i.e., an infinite sequence of elements of A. We denote by $\lambda(i)$ the $(i+1)$-th element of λ. A *segment* π is an element of A^*, i.e., a finite sequence of elements of A.

In our timed model, a trace consists of an infinite sequence of pairs of an instant of time and of the set of actions that are observed at that instant of time. Every such set of actions can be empty so that, on the one hand, we can model components that stop executing after a certain point in time while still part of a system and, on the other hand, we can model observations that are triggered by actions performed by components outside the system.

Definition 1 (Timed traces) *Let A be a set (of actions).*

- *A* time sequence τ *is a trace over* $\mathbb{R}_{\geq 0}$ *such that:*
  - $\tau(0) = 0$*;*
  - *for every* $i \in \mathbb{N}$, $\tau(i) < \tau(i+1)$*;*
  - *the set* $\{\tau(i) : i \in \mathbb{N}\}$ *is unbounded, i.e., time progresses (also called the 'non-Zeno' condition).*
- *An* action sequence σ *is a trace over* 2^A*, i.e., an infinite sequence of sets of actions, such that* $\sigma(0) = \emptyset$*.*
- *A* timed trace *over A is a pair $\lambda = \langle \sigma, \tau \rangle$ of an action and a time sequence. We denote by $\Lambda(A)$ the set of timed traces over A and we call any $\Lambda \subseteq \Lambda(A)$ a* timed property*.*
- *Given* $\delta \in \mathbb{R}_{>0}$*, the* δ-time sequence τ_δ *consists of all multiples of* δ *— for every* $i \in \mathbb{N}$*,* $\tau_\delta(i) = i \cdot \delta$*. A* δ-timed trace *over A is a timed trace* $\langle \sigma, \tau_\delta \rangle$*.*

That is, in δ-timed traces, actions occur according to a fixed period (δ). These traces are useful to capture the behaviour of discrete systems that execute according to a fixed clock granularity.

In order to address heterogeneity, we need a notion of time refinement:

Definition 2 (Time refinement) *Let $\rho : \mathbb{N} \to \mathbb{N}$ be a monotonically increasing function that satisfies $\rho(0) = 0$.*

– *Let τ, τ' be two time sequences. We say that τ'* refines *τ through ρ ($\tau' \preceq_\rho \tau$) iff, for every $i \in \mathbb{N}$, $\tau(i) = \tau'(\rho(i))$. We say that τ'* refines *τ ($\tau' \preceq \tau$) iff $\tau' \preceq_\rho \tau$ for some ρ.*
– *Let $\lambda = \langle \sigma, \tau \rangle$, $\lambda' = \langle \sigma', \tau' \rangle$ be two timed traces. We say that λ' refines λ through ρ ($\lambda' \preceq_\rho \lambda$) iff*
  - *$\tau' \preceq_\rho \tau$,*
  - *$\sigma(i) = \sigma'(\rho(i))$ for every $i \in \mathbb{N}$, and*
  - *$\sigma'(j) = \emptyset$ for every $\rho(i) < j < \rho(i+1)$.*

  We also say that λ' refines *λ ($\lambda' \preceq \lambda$) iff $\lambda' \preceq_\rho \lambda$ for some ρ.*
– *The* r-closure *of a timed property Λ is $\Lambda^r = \{\lambda' : \exists \lambda \in \Lambda (\lambda' \preceq \lambda)\}$.*
– *We say that Λ is* closed under time refinement*, or r-closed, iff $\Lambda^r \subseteq \Lambda$.*

We extend the notion of refinement to timed properties:

– *A timed property Λ' refines a timed property Λ ($\Lambda' \preceq \Lambda$) iff, for every $\lambda' \in \Lambda'$, there exists $\lambda \in \Lambda$ such that $\lambda' \preceq \lambda$.*
– *A timed property Λ'* approximates *a timed property Λ ($\Lambda' \precapprox \Lambda$) iff $\Lambda' \preceq \Lambda$ and, for every $\lambda \in \Lambda$, there exists $\lambda' \in \Lambda'$ such that $\lambda' \preceq \lambda$.*

That is, a time sequence refines another if the former interleaves time observations between any two time observations of the latter. For instance,

$$\langle \emptyset \cdot \{a, b\} \cdot \{b, c\} \ldots, 0 \cdot 2 \cdot 4 \cdots \rangle$$

is refined by

$$\langle \emptyset \cdot \emptyset \cdot \{a, b\} \cdot \emptyset \cdot \{b, c\} \ldots, 0 \cdot 1 \cdot 2 \cdot 3 \cdot 4 \cdots \rangle$$

Refinement extends to traces by requiring that no actions be observed in the finer trace between two consecutive times of the coarser. A timed property Λ' refines Λ if all traces of Λ' refine some trace of Λ. If all the traces of Λ have a refinement in Λ', then Λ' approximates Λ.

2.2 Timed input/output automata

In order to model machines, we use timed I/O automata as in [7] except that transitions perform sets of actions instead of single actions. Working with sets of actions simplifies the treatment of interconnections by introducing synchronisation sets and gives us for free the empty set as an abstraction of actions performed by the environment that an automaton can observe without being directly involved.

A timed automaton is defined in terms of a finite set $\mathbb{C}$ of clocks. A *condition* over $\mathbb{C}$ is a finite conjunction of expressions of the form $c \bowtie n$ where $c \in \mathbb{C}$, $\bowtie \in \{\leq, \geq\}$ and $n \in \mathbb{N}$. We denote by $\mathcal{B}(\mathbb{C})$ the set of conditions over $\mathbb{C}$.

Definition 3 (TIOA) *A timed I/O automaton $\mathcal{A}$ (TIOA) is a tuple*

$$\mathcal{A} = \langle Loc, q_0, \mathbb{C}, E, Act, Inv \rangle$$

where:

- *Loc is a finite set of locations;*
- $q_0 \in Loc$ *is the initial location;*
- $\mathbb{C}$ *is a finite set of clocks;*
- $E \subseteq Loc \times 2^{Act} \times \mathcal{B}(\mathbb{C}) \times 2^{\mathbb{C}} \times Loc$ *is a finite set of edges;*
- $Act = Act^I \cup Act^O \cup Act^\tau$ *is a finite set of actions partitioned into inputs, outputs and internal actions, respectively;*
- Inv: $Loc \to \mathcal{B}(\mathbb{C})$ *is a mapping that associates an invariant with every location.*

In addition, we impose that every TIOA is open *in the sense of not interfering with the ability of the environment to make progress: for all* $l \in Loc$*, there is an edge* $(l, \emptyset, \phi, \emptyset, l') \in E$ *for some location* l' *such that* $Inv(l')$ *is implied by* $Inv(l)$ *and for some tautology* ϕ*.*

Given an edge (l, S, C, R, l'), l is the source location, l' is the target location, S is the set of actions executed during the transition, C is a guard (a condition that determines if the transition can be performed), and R is the set of clocks that are reset by the transition. The requirement that every location is the source of a transition labelled by $\emptyset$ that is always enabled means that the behavior of $\mathcal{A}$ is always open to the execution of actions in which it is not involved.

A *clock valuation* over a set $\mathbb{C}$ of clocks is a mapping v: $\mathbb{C} \to \mathbb{R}_{\geq 0}$. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation v, we denote by $v+d$ the valuation defined by, for any clock $c \in \mathbb{C}$, $(v+d)(c) = v(c)+d$. Given $R \subseteq \mathbb{C}$ and a clock valuation v, we denote by $v^{\mathbf{R}}$ the valuation where clocks from R are reset, i.e., such that $v^{\mathbf{R}}(c)=0$ if $c \in R$ and $v^{\mathbf{R}}(c)=v(c)$ otherwise. Given a condition C in $\mathcal{B}(\mathbb{C})$, we use $v \vDash C$ to express that C holds for the clock valuation v.

Definition 4 (Execution) *Let* $\mathcal{A} = \langle Loc, q_0, \mathbb{C}, E, Act, Inv \rangle$ *be a TIOA. An* execution *of* $\mathcal{A}$ *starting in* l_0 *and valuation* v_0 *is a sequence*

$$(l_0, v_0, d_0) \xrightarrow{S_0, R_0} (l_1, v_1, d_1) \xrightarrow{S_1, R_1} \dots$$

where, for all i*: (1)* $l_i \in Loc$*,* v_i *is a clock valuation over* $\mathbb{C}$ *and* $d_i \in \mathbb{R}_{>0}$*; (2)* $S_i \subseteq Act$ *and* $R_i \subseteq \mathbb{C}$*; (3) for all* $0 \leq t \leq d_i$*,* $v_i + t \vDash Inv(l_i)$*; (4)* $v_{i+1} = (v_i + d_i)^{\mathbf{R_i}}$*; and (5) there is* $(l_i, S_i, C, R_i, l_{i+1}) \in E$ *such that* $v_i + d_i \vDash C$*. A* partial execution *is of the form*

$$(l_0, v_0, d_0) \xrightarrow{S_0, R_0} \dots \xrightarrow{S_{n-1}, R_{n-1}} (l_n, v_n, d_n)$$

where (1) and (3) hold for all $i \in [0..n]$*, and (2), (4) and (5) for all* $i \in [0..n-1]$*.*

That is, each triple (l_i, v_i, d_i) consists of a location, the value of the clocks when that location is reached at that point of the execution, and the duration for which the automaton remains at that location before the next transition (which can leave the automaton in the same location). During this time, the invariant $Inv(l_i)$ must hold. A transition out of (l_i, v_i, d_i) happens at the end of d_i units of time and needs to be made by an edge whose guard C_i holds at that time and leads to a location whose invariant is satisfied. As a result of the transition, the clocks are updated to $(v_i + d_i)^{\mathbf{R_i}}$.

A pair (l, v) where l is a location and v is a clock valuation is said to be *reachable at time* $T \in \mathbb{R}_{\geq 0}$ if either (a) $(l, v) = (q_0, 0)$, $T = 0$ and, there exists $d_0 > 0$ such that $t \vDash Inv(q_0)$ for all $0 \leq t \leq d_0$; or (b) there exists a partial execution that starts at $(q_0, 0)$ and ends at $(l_n, v_n) = (l, v)$, and $T = \sum_{i=0 \cdots n-1} d_i$.

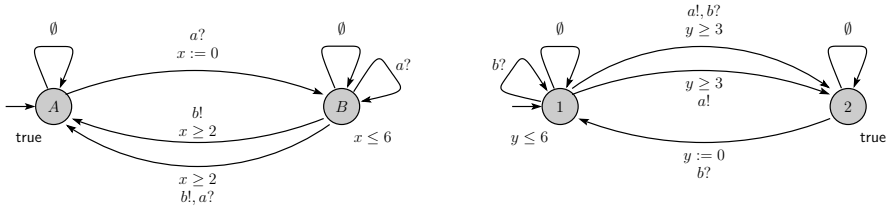

Fig. 1. Two TIOAs: $\mathcal{A}^x$ (left) and $\mathcal{A}^y$ (right)

Example 5 *Consider the TIOAs in Fig. 1:* $\mathcal{A}^x = \langle \{A, B\}, A, \{x\}, E^x, Act_x, Inv_x \rangle$ *with* $Act_x^I = \{a\}$ *and* $Act_x^O = \{b\}$*, and* $\mathcal{A}^y = \langle \{1, 2\}, 1, \{y\}, E^y, Act_y, Inv_y \rangle$ *with* $Act_y^I = \{b\}$ *and* $Act_y^O = \{a\}$ *(for clarity, inputs are decorated with ? and outputs with !).*

- $\mathcal{A}^y$ *starts by sending an* a *within six time units but not before three units have passed; it then waits for receiving a* b *to start again and send another* a*. More* b*'s can be received meanwhile (even while sending an* a*), but they are all ignored.*
- $\mathcal{A}^x$ *waits for receiving an* a*, after which it sends a* b *within six time units but not before two times units have passed (all* a*'s received in the meanwhile being ignored); then,* $\mathcal{A}^x$ *waits for receiving another* a*.*

An example of a partial execution of $\mathcal{A}^x$ *is*

$$(A, 0, 2) \xrightarrow{\{a\},\{x\}} (B, 0, 3) \xrightarrow{\{b\},\emptyset} (A, 3, 5) \xrightarrow{\{a\},\{x\}} (B, 0, 2)$$

which shows that $(B, 0)$ *is reachable at times* 2 *and* 10*.*

We now recall the classical definition of composition of *compatible* TIOAs, which captures partial synchronisation.

Definition 6 (Compatibility) *Two TIOAs* $\mathcal{A}_i = \langle Loc^i, q_0^i, \mathbb{C}^i, E^i, Act_i, Inv^i \rangle$ *are compatible iff* $\mathbb{C}^1 \cap \mathbb{C}^2 = Act_1^I \cap Act_2^I = Act_1^O \cap Act_2^O = Act_1^\tau \cap Act_2 = Act_2^\tau \cap Act_1 = \emptyset$*.*

Definition 7 (Composition) *The* composition *of two compatible TIOAs* $\mathcal{A}_i = \langle Loc^i, q_0^i, \mathbb{C}^i, E^i, Act_i, Inv^i \rangle$ *is*

$$\mathcal{A}_1 \parallel \mathcal{A}_2 = \langle Loc^1 \times Loc^2, (q_0^1, q_0^2), \mathbb{C}^1 \cup \mathbb{C}^2, E, Act, Inv \rangle$$

where:

- $Act^I = (Act_1^I \backslash Act_2^O) \cup (Act_2^I \backslash Act_1^O)$,
- $Act^O = (Act_1^O \backslash Act_2^I) \cup (Act_2^O \backslash Act_1^I)$,
- $Act^\tau = Act_1^\tau \cup Act_2^\tau \cup (Act_1^I \cap Act_2^O) \cup (Act_1^O \cap Act_2^I)$, *and*
- *for all* $(q_1, q_2) \in Loc^1 \times Loc^2$*:*
  - $Inv((q_1, q_2)) = Inv^1(q_1) \wedge Inv^2(q_2)$*;*
  - $((q_1, q_2), S, C, R, (q_1', q_2')) \in E$ *iff* $(q_1, S_1, C_1, q_1') \in E^1$, $(q_2, S_2, C_2, q_2') \in E^2$, $C = C_1 \wedge C_2$, $S_i = S \cap Act_i$ *(*$i = 1, 2$*) and* $R = R_1 \cup R_2$.

Notice that, because the guards of transitions are conjoined, for the TIOA that results from the composition to be open (cf. Def. 3) we need to require the existence of a transition labeled with $\emptyset$ and a tautological guard (instead of simply *true*). Notice also that, by construction, whenever $S \cap Act_1 \neq \emptyset$ and $S \cap Act_2 \neq \emptyset$, all actions on which $\mathcal{A}_1$ and $\mathcal{A}_2$ synchronise (those in $S \cap Act_1 \cap Act_2$) are necessarily inputs on one side and outputs on the other; the composition makes those actions internal. Finally, transitions such that $S \cap Act_i = \emptyset$, which are usually considered as non-synchronising, are in our case handled as synchronising transitions where $\mathcal{A}_i$ performs the empty set of actions (which corresponds to an open semantics).

3 Timed machines: definition and operations

In order to model systems where applications execute over specific platforms, which implies that they are subject to the clock granularity of the platform, we extend TIOAs to what we call timed machines.

3.1 Timed machines

A timed machine is a TIOA that executes in the context of a clock granularity δ, i.e., its actions are always executed at instant times that are multiples of δ.

Definition 8 (DTIOM) *A* discrete timed I/O machine *(DTIOM) is a pair*

$$\mathcal{M} = \langle \delta_{\mathcal{M}}, \mathcal{A}_{\mathcal{M}} \rangle$$

where $\delta_{\mathcal{M}} \in \mathbb{R}_{>0}$ *and* $\mathcal{A}_{\mathcal{M}}$ *is a TIOA.*

The executions *and* partial executions *of* $\mathcal{M}$ *are those of* $\mathcal{A}_{\mathcal{M}}$ *restricted to transitions at every* $\delta_{\mathcal{M}}$*, i.e.,*

$$(l_0, v_0, d_0) \xrightarrow{S_0, R_0} (l_1, v_1, d_1) \xrightarrow{S_1, R_1} \ldots$$

such that all the durations d_i *are* $\delta_{\mathcal{M}}$*. Therefore, we represent executions of DTIOMs as sequences*

$$(l_0, v_0) \xrightarrow{S_0, R_0} (l_1, v_1) \xrightarrow{S_1, R_1} \ldots$$

and call each pair (l_i, v_i) *an execution state.*

The behaviour $[\![\mathcal{M}]\!]$ *of* $\mathcal{M}$ *is the set of executions such that* $l_0{=}q_0$ *and* $v_0(c){=}0$ *for all* $c{\in}\mathbb{C}$*, i.e., those that start in the initial location with all clocks set to* 0*.*

Every execution of a DTIOM $\mathcal{M}$ *defines the* $\delta_{\mathcal{M}}$*-timed trace* $\lambda{=}\langle\sigma, \tau_{\delta_{\mathcal{M}}}\rangle$ *over Act where* $\sigma(0){=}\emptyset$ *and, for* $i \geq 0$*,* $\sigma(i+1) = S_i$*. We denote by* $\Lambda_{\mathcal{M}}$ *the r-closure of the set of timed traces defined by* $[\![\mathcal{M}]\!]$*, which we call its* language*.*

The fact that the language of a DTIOM is r-closed means that it contains all possible interleavings of empty observations, thus capturing the behaviour of the DTIOM in any possible environment. This notion of closure can be related to mechanisms that, such as stuttering [1], ensure that components do not constrain their environment.

Example 9 *Consider* $\mathcal{M}^x = \langle\delta_x, \mathcal{A}^x\rangle$*, and* $\mathcal{M}^y = \langle\delta_y, \mathcal{A}^y\rangle$*, with* $\delta_x = 2$*,* $\delta_y = 1$ *and* $\mathcal{A}^x$ *and* $\mathcal{A}^y$ *as in Ex. 5. Notice that the partial execution of* $\mathcal{A}^x$ *given in Ex. 5 is not a partial execution of* $\mathcal{M}^x$ *as it does not respect the granularity* $\delta_x = 2$*. An example of a partial execution of* $\mathcal{M}^x$ *is*

$$(A,0) \xrightarrow{\{a\},\{x\}} (B,0) \xrightarrow{\emptyset,\emptyset} (B,2) \xrightarrow{\{b\},\emptyset} (A,4)$$

Note that this means that a *was executed at time* 2*, nothing was executed at time* 4 *and* b *was executed at time* 6*.*

3.2 Composition and refinement of timed machines

Composition of DTIOMs with the same clock granularity is as for TIOA:

Definition 10 (Composition) *Given two TIOAs* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *that are compatible, we define the composition* $\langle\delta, \mathcal{A}_1\rangle \| \langle\delta, \mathcal{A}_2\rangle = \langle\delta, \mathcal{A}_1 \| \mathcal{A}_2\rangle$*.*

It is not difficult to prove that the language $\Lambda_{\langle\delta,\mathcal{A}_1\|\mathcal{A}_2\rangle}$ of the composition is the intersection

$$\Lambda^{\iota_1}_{\langle\delta,\mathcal{A}_1\rangle} \cap \Lambda^{\iota_2}_{\langle\delta,\mathcal{A}_2\rangle}$$

where

- ι_i is the inclusion of Act_i in $Act_1 \cup Act_2$ and
- $\Lambda^{\iota_i}_{\langle\delta,\mathcal{A}_i\rangle} = \{\lambda : \iota_i^{-1}(\lambda) \in \Lambda_{\langle\delta,\mathcal{A}_i\rangle}\}$ are the projections of the languages of the machines to the alphabet of the composition defined by, for every k, $\iota_i^{-1}(\lambda)(k) = \iota_i^{-1}(\lambda(k))$.

That is, the language of the composition consists of the timed traces that project to timed traces of languages of the component DTIOMs. This is what is usually taken to be the joint behaviour of a system of components in a trace-based semantic domain, meaning that $\langle\delta, \mathcal{A}_1 \| \mathcal{A}_2\rangle$ provides a model of the joint behaviour of two systems of which $\langle\delta, \mathcal{A}_1\rangle$ and $\langle\delta, \mathcal{A}_2\rangle$ are models.

If $\langle\delta_1, \mathcal{A}_1\rangle$ and $\langle\delta_2, \mathcal{A}_2\rangle$ have different clock granularities, we can still calculate the intersection $\Lambda^{\iota_1}_{\langle\delta,\mathcal{A}_1\rangle} \cap \Lambda^{\iota_2}_{\langle\delta,\mathcal{A}_2\rangle}$, which is the joint behaviour of the two machines synchronising on shared inputs and outputs at times that are multiple of both

δ_1 and δ_2. If no such multiples exist, the two machines cannot synchronise and, therefore, either they do not have liveness requirements, in which case they can agree on timed traces that only execute the empty set of actions, or they cannot agree on any timed trace — their interconnection is inconsistent.

If δ_1 and δ_2 admit a common multiple, i.e., $\delta_1 \cdot n = \delta_2 \cdot m$ for given $n, m \in \mathbb{N}_{>0}$, then they are *commensurable*, i.e., they admit a common divisor ($\delta_1/m = \delta_2/n$) – again, a real number. This is the situation that we characterise now. More precisely, our aim is to construct a machine $\mathcal{M}$ that, although it may not generate the full set of joint behaviours, i.e., be such that $\Lambda_{\mathcal{M}} = \Lambda^{\iota_1}_{\mathcal{M}_1} \cap \Lambda^{\iota_2}_{\mathcal{M}_2}$, it will be the 'best' approximation of that set in the sense that $\Lambda_{\mathcal{M}} \precapprox \Lambda^{\iota_1}_{\mathcal{M}_1} \cap \Lambda^{\iota_2}_{\mathcal{M}_2}$ and, for any other machine $\mathcal{M}'$ such that $\Lambda_{\mathcal{M}'} \precapprox \Lambda^{\iota_1}_{\mathcal{M}_1} \cap \Lambda^{\iota_2}_{\mathcal{M}_2}$, $\Lambda_{\mathcal{M}'} \precapprox \Lambda_{\mathcal{M}}$. Having a best approximation is important so that properties of the global behaviour of the system (such as consistency and feasibility, discussed in Sec. 4) can be inferred from that of the composed machine or that the behaviour of the system can be simulated through a machine.

The idea is to refine the timed machines to a common clock granularity and then compose the refinements: intuitively, given a timed machine $\mathcal{M} = \langle \delta, \mathcal{A} \rangle$, we define its *k-refinement* $\mathcal{M}_k = \langle \delta/k, \mathcal{A}_k \rangle$ by dividing both the clock granularity and the TIOA $\mathcal{A}$ by k so as to produce a TIOA $\mathcal{A}_k$ that divides every state in k copies such that the original transitions are performed in the last 'tick', all previous 'ticks' performing no actions and, therefore, being open for synchronisation with a machine that ticks with a granularity δ/k.

Definition 11 (Refinement) *Given a TIOA $\mathcal{A} = \langle Loc, q_0, \mathbb{C}, E, Act, Inv \rangle$ and $k \in \mathbb{N}_{>0}$, its* k-refinement *is the TIOA $\mathcal{A}_k = \langle Loc_k, q_{k0}, \mathbb{C}, E_k, Act, Inv_k \rangle$ where:*

- $Loc_k = Loc \times [0..k-1]$;
- $q_{k0} = (q_0, 0)$;
- $Inv_k(l, i) = Inv(l)$;
- *for every (l, S, C, R, l') of E, E_k has the edge $((l, k-1), S, C, R, (l', 0))$ and all edges of the form $((l, i), \emptyset, true, \emptyset, (l, i+1))$, $i \in [0..k-2]$.*

It is easy to see that $\Lambda_{\mathcal{M}_k} \preceq \Lambda_{\mathcal{M}}$, i.e., the language of $\mathcal{M}_k$ refines that of $\mathcal{M}$; in fact, because the languages are r-closed, $\Lambda_{\mathcal{M}_k} \subseteq \Lambda_{\mathcal{M}}$. Because every execution of $\mathcal{M}$ defines a (unique) execution of $\mathcal{M}_k$, the language of $\mathcal{M}_k$ actually approximates that of $\mathcal{M}$, i.e., $\Lambda_{\mathcal{M}_k} \precapprox \Lambda_{\mathcal{M}}$, meaning that all possible behaviours of $\mathcal{M}$ can be accounted for in $\mathcal{M}_k$ through a refinement: we say that $\mathcal{M}_k$ *approximates* $\mathcal{M}$ and write $\mathcal{M}_k \precapprox \mathcal{M}$. More generally, for arbitrary DTIOMs $\mathcal{M}$ and $\mathcal{M}'$ that have a common alphabet (i.e., $Act_{\mathcal{M}'} = Act_{\mathcal{M}}$), we define $\mathcal{M}' \precapprox \mathcal{M}$ to mean that $\delta_{\mathcal{M}}$ is a multiple of $\delta_{\mathcal{M}'}$ and $\Lambda_{\mathcal{M}'} \precapprox \Lambda_{\mathcal{M}}$.

Example 12 *Consider the TIOA $\mathcal{A}^x$ in Fig. 1 and the corresponding DTIOM $\mathcal{M}^x$ defined in Ex. 9, which has granularity 2. Its refinement to a DTIOM with granularity 1 is $\mathcal{M}^x_2 = \langle 1, \mathcal{A}^x_2 \rangle$, with $\mathcal{A}^x_2$ given in Fig. 2. The refinement of the partial execution of $\mathcal{M}^x$ given in Ex. 9 is:*

$$((A,0),0) \xrightarrow{\emptyset,\emptyset} ((A,1),1) \xrightarrow{\{a\},\{x\}} ((B,0),0) \xrightarrow{\emptyset,\emptyset} ((B,1),1) \xrightarrow{\emptyset,\emptyset} ((B,0),2) \xrightarrow{\emptyset,\emptyset} ((B,1),3) \xrightarrow{\{b\},\emptyset} ((A,0),4)$$

We can now extend the composition of two timed machines to the case where their clock granularities are commensurable (have a common divisor):

Definition 13 (Heterogeneous compatibility) *Two DTIOMs* $\mathcal{M}_i = \langle \delta_i, \mathcal{A}_i \rangle$, $i = 1, 2$, *are said to be* δ-compatible *(where* $\delta \in \mathbb{R}_{>0}$*) if (a)* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are compatible, and (b)* δ *is a common divisor of* δ_1 *and* δ_2*. They are said to be* compatible *if they are* δ-compatible *for some* δ*.*

Definition 14 (Heterogeneous composition) *The* δ-composition *of two* δ-compatible *DTIOMs is*

$$\mathcal{M}_1 \parallel_\delta \mathcal{M}_2 = \mathcal{M}_{1(\delta_1/\delta)} \parallel \mathcal{M}_{2(\delta_2/\delta)} = \langle \delta, \mathcal{A}_{1(\delta_1/\delta)} \parallel \mathcal{A}_{2(\delta_2/\delta)} \rangle$$

If δ *is the greatest common divisor of* δ_1 *and* δ_2*, we use the notation* $\mathcal{M}_1 \| \mathcal{M}_2$ *and simply refer to* the composition *of* $\mathcal{M}_1$ *and* $\mathcal{M}_2$*.*

Notice that if $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible, so are $\mathcal{A}_{1(\delta_1/\delta)}$ and $\mathcal{A}_{2(\delta_2/\delta)}$.

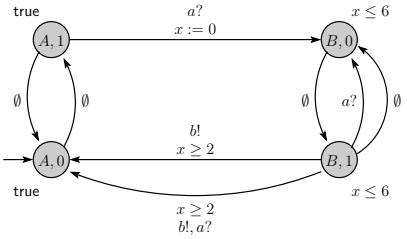

Fig. 2. The refinement $\mathcal{A}_2^x$ of $\mathcal{A}^x$

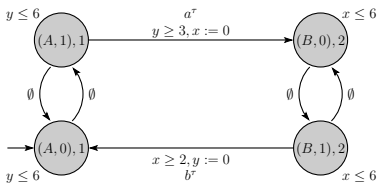

Fig. 3. The TIOA $\mathcal{A}^{x,y}$ of $\mathcal{M}^x \| \mathcal{M}^y$

Example 15 *Consider DTIOMs* $\mathcal{M}^x$ *and* $\mathcal{M}^y$ *from Ex. 9. Because* $\mathcal{A}^x$ *and* $\mathcal{A}^y$ *are compatible and* δ_x *and* δ_y *have a common divisor (*$\delta = 1$*), we can compute their composition. The first step consists in refining* $\mathcal{A}^x$ *into* $\mathcal{A}_2^x$ *(Fig. 2). The composition* $\mathcal{M}^x \| \mathcal{M}^y$ *is* $\langle 1, \mathcal{A}^{x,y} \rangle$ *where* $\mathcal{A}^{x,y} = \mathcal{A}_2^x \| \mathcal{A}^y$ *is given in Fig. 3. Notice that actions* a *and* b *are synchronised and, hence, made internal in the composition, which we denote by* a^τ *and* b^τ*, respectively.*

The language of a heterogeneous composition is not necessarily the intersection of the languages of the components. However, if $\mathcal{M}_1$ and $\mathcal{M}_2$ can be composed, the machine $\mathcal{M}_1 \| \mathcal{M}_2$ approximates $\Lambda^{\iota_1}_{\mathcal{M}_1} \cap \Lambda^{\iota_2}_{\mathcal{M}_2}$, and is a best approximation:

Theorem 16 *Let* $\mathcal{M}_i$*,* $i = 1, 2$*, be two compatible DTIOMs. The composition* $\mathcal{M}_1 \| \mathcal{M}_2$ *is the machine that best approximates* $\Lambda = \Lambda^{\iota_1}_{\mathcal{M}_1} \cap \Lambda^{\iota_2}_{\mathcal{M}_2}$*, i.e.,*

- $\Lambda_{\mathcal{M}_1 \| \mathcal{M}_2} \precapprox \Lambda$ *and,*
- *for any other machine* $\mathcal{M}$ *such that* $\Lambda_{\mathcal{M}} \precapprox \Lambda$*,* $\mathcal{M} \precapprox \mathcal{M}_1 \| \mathcal{M}_2$*.*

3.3 Büchi representation of timed machines

In order to check different structural properties of DTIOMs, namely properties formulated in terms of reachable states, it is useful to be able to construct Büchi-automata "equivalent".

Let $\mathcal{A} = \langle Loc, l_0, \mathbb{C}, E, Act, Inv \rangle$ be a TIOA. Given a clock c, let $\mathsf{Max}^{\mathcal{A}}(c)$ denote the maximal constant with which c is compared in the guards and invariants of $\mathcal{A}$. Let $\mathcal{M} = \langle \delta, \mathcal{A} \rangle$ and $\mathcal{B}_{\mathcal{M}} = \langle Q, q_0, 2^{Act}, \rightarrow, Q \rangle$ be the Büchi automaton such that:

- $Q = Loc \times \prod_{c \in \mathbb{C}} [0 \mathinner{..} \lfloor \frac{\mathsf{Max}^{\mathcal{M}}(c)}{\delta} \rfloor + 1]$ (i.e., states consist of a location l and a natural number $n_c \leq \lfloor \frac{\mathsf{Max}^{\mathcal{M}}(c)}{\delta} \rfloor + 1$, for every $c \in \mathbb{C}$);
- $q_0 = (l_0, \mathbf{0})$;
- $(l, \boldsymbol{\nu}) \xrightarrow{S} (l', \boldsymbol{\nu}')$ iff there exists a transition $(l, S, C, R, l') \in E$ such that:
  - (i) for all $0 \leq t \leq \delta$, $\boldsymbol{\nu} \cdot \delta + t \models Inv(l)$,
  - (ii) $\boldsymbol{\nu} \cdot \delta + \delta \models C$,
  - (iii) for all $c \in \mathbb{C}$, $\boldsymbol{\nu}'(c) = \left\{ \begin{array}{ll} 0 & \text{if } c \in R \\ \boldsymbol{\nu}(c) & \text{if } c \notin R \text{ and } \boldsymbol{\nu}(c) = \lfloor \frac{\mathsf{Max}^{\mathcal{A}}(c)}{\delta} \rfloor + 1 \\ \boldsymbol{\nu}(c) + 1 & \text{otherwise} \end{array} \right\}$
  - (iv) $\boldsymbol{\nu}' \cdot \delta \models Inv(l')$.

Notice that Q involves only natural numbers. The size of $\mathcal{B}_{\mathcal{M}}$ is in $O(|Loc| \cdot (\lfloor \frac{\mathsf{Max}}{\delta} \rfloor + 2)^{|\mathbb{C}|})$, where $|Loc|$ and $|\mathbb{C}|$ are the size of Loc and the number of clocks, respectively, and $\mathsf{Max} = \max\{\mathsf{Max}^{\mathcal{A}}(c) \mid c \in \mathbb{C}\}$ is the maximal constant considered in all constraints and invariants of $\mathcal{M}$.

The Büchi automaton $\mathcal{B}_{\mathcal{M}}$ is equivalent to $\mathcal{M}$ in the following sense:

Theorem 17 *For all action sequences σ over Act, $\langle \sigma, \tau_\delta \rangle \in [\![\mathcal{M}]\!]$ iff the infinite sequence $\sigma(1)\sigma(2)\ldots$ is in the language of $\mathcal{B}_{\mathcal{M}}$.*

4 Consistency and feasibility of timed machines

In this section, we investigate two important properties of DTIOMs as models of systems: consistency (in the sense that they generate a non-empty language) and feasibility (in the sense that they generate a non-empty language no matter what inputs they receive). We are especially interested in conditions under which consistency/feasibility are preserved by composition. This is because, in order to support run-time interconnections, one should be able to guarantee that a composition of DTIOMs is consistent/feasible *without* having to compose them.

4.1 Consistency

Definition 18 (Consistency) *A DTIOM $\mathcal{M}$ is said to be* consistent *if $\Lambda_{\mathcal{M}} \neq \emptyset$.*

Notice that consistency is preserved by refinement:

Proposition 19 *Let $k \in \mathbb{N}_{>0}$. A DTIOM $\mathcal{M}$ is consistent iff its k-refinement $\mathcal{M}_k$ is consistent. More generally, for arbitrary DTIOM $\mathcal{M}$ and $\mathcal{M}'$,*

- *if $\mathcal{M}' \preceq \mathcal{M}$ and $\mathcal{M}'$ is consistent, then so is $\mathcal{M}$, and*
- *if $\mathcal{M}' \precapprox \mathcal{M}$, then $\mathcal{M}'$ is consistent iff $\mathcal{M}$ is consistent.*

A sufficient condition for consistency is that the DTIOM is initializable and makes independent progress. A DTIOM is initializable if it can stay in the initial state until the first tick of the clock:

Definition 20 (Initializable) *A DTIOM $\mathcal{M}$ is said to* be initializable *if, for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(q_0, t) \vDash Inv(q_0)$.*

For a machine to make independent progress (which we adapt from [7]), it needs to make a transition from any reachable state without forcing the environment to provide any input:

Definition 21 (Independent progress) *A DTIOM $\mathcal{M}$ is said to* make independent progress *if, for every reachable state (l, v), there is an edge (l, A, C, R, l') such that: (a) $A \subseteq Act^{O}_{\mathcal{M}} \cup Act^{\tau}_{\mathcal{M}}$, (b) $v + \delta_{\mathcal{M}} \vDash C$, and (c) for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}} + t \vDash Inv(l')$.*

As an example, both $\mathcal{M}^x$ and $\mathcal{M}^y$ as in Ex. 9 are initializable and make independent progress.

Proposition 22 *Any initializable DTIOM that makes independent progress is consistent.*

Notice that checking that a timed machine makes independent progress requires only the analysis of properties of its reachable states. In practice, this can be done using a syntactic check on the Büchi automaton as constructed in Sec. 3.3: a given DTIOM $\mathcal{M}$ makes independent progress iff all reachable states $(l, \boldsymbol{\nu})$ of the equivalent Büchi automaton $\mathcal{B}_{\mathcal{M}}$ have at least one outgoing transition $(l, \boldsymbol{\nu}) \xrightarrow{A} (l', \boldsymbol{\nu}')$ with $A \subseteq Act^{O}_{\mathcal{M}} \cup Act^{\tau}_{\mathcal{M}}$. $\mathcal{B}_{\mathcal{M}}$ has only finitely many states, denoted by $|\mathcal{B}_{\mathcal{M}}|$, and finitely many transitions, denoted by $|E_{\mathcal{M}}|$, and, hence, this can be checked in time $O(|\mathcal{B}_{\mathcal{M}}| \cdot |E_{\mathcal{M}}|)$.

4.2 Compositional consistency checking

In order to investigate conditions that can guarantee compositionality of consistency checking, we start by remarking that the fact that two DTIOMs $\mathcal{M}_1$ and $\mathcal{M}_2$ are such that δ_1 and δ_2 are commensurate simply means that we can find a clock granularity in which we can accommodate the transitions that the two DTIOMs perform: by itself, this does not ensure that the two DTIOMs can jointly execute their input/output synchronisation pairs. For example, if $\delta_1 = 2$ and $\delta_2 = 3$ and $\mathcal{M}_2$ only performs non-empty actions at odd multiples of 3, the two machines will not be able to agree on their input/output synchronisation pairs. For the DTIOMs to actually interact with each other it is necessary that their input/output synchronisation pairs can be performed on a common multiple of δ_1 and δ_2.

Definition 23 (Cooperative) *A DTIOM $\mathcal{M}$ is said to be* cooperative in relation to $Q \subseteq Act_{\mathcal{M}}$ *and a multiple δ of $\delta_{\mathcal{M}}$ if the following holds for every (l, v) reachable at a time T such that $(T + \delta_{\mathcal{M}})$ is not a multiple of δ:*

> *for every edge $(l, A, C, R, l') \in E_{\mathcal{M}}$ such that $v + \delta_{\mathcal{M}} \vDash C$ and $(v + \delta_{\mathcal{M}})^{\mathbf{R}} + t \vDash Inv_{\mathcal{M}}(l')$ for all $0 \leq t \leq \delta_{\mathcal{M}}$ — i.e., the machine makes a transition at a time that is not a multiple of δ — there exists an edge $(l, A \backslash Q, C', R', l'')$ such that $v + \delta_{\mathcal{M}} \vDash C'$; for all $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}'} + t \vDash Inv_{\mathcal{M}}(l'')$ — i.e., the machine can make an alternative transition that does not perform any actions in Q.*

Essentially, being cooperative in relation to Q and δ means that the machine will not force transitions that perform actions in Q at times that are not multiples of δ. In practice, this can be verified using a syntactic check on the states of the equivalent Büchi automaton that can be reached with a number of transitions n such that $n + 1$ is not a multiple of $\delta/\delta_{\mathcal{M}}$. This can be done in time in $O(\frac{\delta}{\delta_{\mathcal{M}}} \cdot |\mathcal{B}_{\mathcal{M}}| \cdot |E_{\mathcal{M}}|^2)$, with $|\mathcal{B}_{\mathcal{M}}|$ the size of the Büchi automaton $\mathcal{B}_{\mathcal{M}}$ defined in Sec. 3.3.

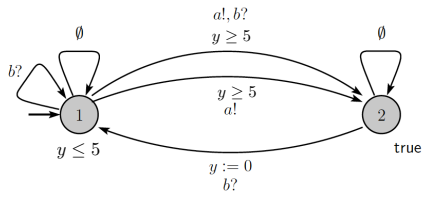

Fig. 4. The TIOA $\mathcal{A}'$.

Example 24 *$\mathcal{M}^y$ from Ex. 9 is cooperative in relation to $\{a, b\}$ and $\delta = 2$. In constrast, the machine $\mathcal{M}'$ with $\delta' = 1$ and the TIOA $\mathcal{A}'$ presented in Fig. 4 is not cooperative in relation to $\{a, b\}$ and $\delta = 2$. Indeed, the fact that the state corresponding to the location 1 is reached at time 4 enables the transition $(1, a, y \geq 5, \emptyset, 2)$, which cannot be replaced by $(1, \emptyset, \mathsf{true}, \emptyset, 1)$ because the last condition — for all $0 \leq t \leq 1 = \delta_y$, $5 + t \leq 5$ — is violated. Because the machine $\mathcal{M}'$ forces the output of* a *at time 5, it is easy to conclude that its composition with the machine $\mathcal{M}^x$ from Ex. 9 (which has a clock granularity $\delta^x = 2$) results in a inconsistent DTIOM.*

In relation to the composition of $\mathcal{M}_1$ and $\mathcal{M}_2$, the idea is to require that a common multiple of δ_1 and δ_2 exists such that both DTIOMs are cooperative in relation to $Act_{\mathcal{M}_1} \cap Act_{\mathcal{M}_2}$. However, this is not enough to guarantee that the two DTIOMs can actually work together: we need to ensure that if, say, $\mathcal{M}_1$ wants to output an action, $\mathcal{M}_2$ can accept it.

Definition 25 (DP-enabled) *A DTIOM $\mathcal{M}$ is said to be* DP-enabled in relation to $J \subseteq Act^I_{\mathcal{M}}$ *and* δ *multiple of* $\delta_{\mathcal{M}}$ *if the following property holds for every* $B \subseteq J$ *and state* (l, v) *reachable at a time* T *such that* $(T + \delta_{\mathcal{M}})$ *is a multiple of* δ*:*

> *for every edge* $(l, A, C, R, l') \in E_{\mathcal{M}}$ *such that* $v + \delta_{\mathcal{M}} \vDash C$ *and, for all* $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}} + t \vDash Inv_{\mathcal{M}}(l')$ *— i.e., the machine can make a transition — there exists an edge* $(l, B \cup (A \setminus J), C', R', l'')$ *such that* $v + \delta_{\mathcal{M}} \vDash C'$ *and, for all* $0 \leq t \leq \delta_{\mathcal{M}}$, $(v + \delta_{\mathcal{M}})^{\mathbf{R}'} + t \vDash Inv_{\mathcal{M}}(l'')$ *— i.e., the machine can make an alternative transition that accepts instead* B *as inputs and still performs the same outputs (and inputs outside* J*).*

That is, a DTIOM is DP-enabled in relation to a set of inputs J and a multiple δ of its clock granularity if, whenever it leaves a reachable state at a multiple of δ, it can do so by accepting any subset of J, and if its outputs are independent of the inputs in J that it receives. Both $\mathcal{M}^x$ and $\mathcal{M}^y$ from Ex. 9 are DP-enabled in relation to the set of input actions (resp., $\{a\}$ and $\{b\}$) and $\delta_x = 2$.

Notice that being DP-enabled is different from being input-enabled [14] in that we work with sets of actions (synchronisation sets), not just individual actions in the edges. Because, in our case, inputs and outputs can occur simultaneously, we need to ensure that there is no dependency between those that are included in the same synchronisation set.

DP-enabledness can be verified using a syntactic check on states of the equivalent Büchi automaton that can be reached in a number of steps n such that $n + 1$ is a multiple of $\delta/\delta_{\mathcal{M}}$. This can be done in $O(\frac{\delta}{\delta_{\mathcal{M}}} \cdot |\mathcal{B}_{\mathcal{M}}| \cdot |E_{\mathcal{M}}|^2 \cdot 2^{|Act^I_{\mathcal{M}}|})$, with $|\mathcal{B}_{\mathcal{M}}|$ as given in Sec. 3.3.

We now investigate how a composition can be shown to be consistent. We start by analysing how properties behave under refinement and composition.

Lemma 26 *If a DTIOM $\mathcal{M}$ is initializable (makes independent progress, is DP-enabled / cooperative in relation to J and δ'), then so does $\mathcal{M}_k$ for all $k \in \mathbb{N}_{>0}$.*

That is, refinement preserves initializability, independent progress, DP-enabledness and cooperativeness.

Lemma 27 *Let $\mathcal{M}_i = \langle \delta_i, \mathcal{A}_i \rangle$, $i = 1, 2$, be two δ-compatible DTIOMs and δ'_1 a multiple of δ_1.*

(a) If $\mathcal{M}_1$ and $\mathcal{M}_2$ are initializable so is $\mathcal{M}_1 \parallel_\delta \mathcal{M}_2$.
(b) If $\mathcal{M}_1$ is DP-enabled in relation to $J \subseteq Act^I_1$ and δ'_1, then $\mathcal{M}_1 \parallel_\delta \mathcal{M}_2$ is DP-enabled in relation to $J \setminus Act^O_2$ and δ'_1.
(c) If $\mathcal{M}_1$ is cooperative in relation to $Q \subseteq Act^O_1 \setminus Act^I_2$ and δ'_1, then $\mathcal{M}_1 \parallel_\delta \mathcal{M}_2$ is cooperative in relation to Q and δ'_1.

Notice that in the preservation of DP-enabledness, we need to remove from J any actions that were used for synchronisation with $\mathcal{M}_2$, which are necessarily in Act^O_2. This is because they become internal to the composition and, therefore, are no longer available for synchronisation. The preservation of cooperativeness is relative to set of actions that are not used for synchronisation.

Theorem 28 (Compositionality) *Let $\mathcal{M}_i$, $i = 1, 2$, be two δ-compatible and initializable DTIOMs that can make independent progress. Let δ' be a common multiple of δ_1 and δ_2. If $\mathcal{M}_1$ is DP-enabled in relation to $Act_1^I \cap Act_2^O$ and δ', $\mathcal{M}_2$ is DP-enabled in relation to $Act_2^I \cap Act_1^O$ and δ', and both $\mathcal{M}_1$ and $\mathcal{M}_2$ are δ'-cooperative in relation to $Act_1 \cap Act_2$, then $\mathcal{M}_1 \parallel_\delta \mathcal{M}_2$ is initializable and can make independent progress (and, hence, by Prop. 22, is consistent).*

This result allows us to conclude that the machines $\mathcal{M}^x$ and $\mathcal{M}^y$ presented in Ex. 9 can work together (i.e., $\mathcal{M}^x \parallel \mathcal{M}^y$ is consistent). This is because, as noted before, $\mathcal{M}^x$ and $\mathcal{M}^y$ are DP-enabled in relation to $\delta' = 2$ and $\{a\}$ and $\{b\}$, respectively, and are cooperative in relation to $\{a, b\}$ and $\delta' = 2$.

Notice that, from Lemma 27, if $\mathcal{M}_i$ is DP-enabled in relation to $J \subseteq Act_i^I$ and δ_i' is a multiple of δ_i, the composition $\mathcal{M}_1 \parallel_\delta \mathcal{M}_2$ is DP-enabled in relation to $J \setminus Act_{\overline{i}}^O$ and δ_i', with $\overline{1} = 2$ and $\overline{2} = 1$. Moreover, if $\mathcal{M}_i$ is cooperative in relation to $Q \subseteq Act_i^O \setminus Act_{\overline{i}}^I$ and δ_i' multiple of δ_i, $\mathcal{M}_1 \parallel_\delta \mathcal{M}_2$ is cooperative in relation to Q and δ_i' (also a multiple of δ). This implies that, in order to ensure that the composition of $\mathcal{M}_1 \parallel_\delta \mathcal{M}_2$ with a third machine $\mathcal{M}_3$ (which can itself be the result of a composition) is consistent, we can verify the required properties (being initializable, DP-enabled and cooperative) over the component machines: we do not need to make checks over the machines resulting from the compositions (compositionality).

This result is also important to certify that the behaviour of a system of interacting components — $\Lambda = \Lambda_{\mathcal{M}_1}^{\iota_1} \cap \Lambda_{\mathcal{M}_2}^{\iota_2}$ in the case of two components that implement $\mathcal{M}_1$ and $\mathcal{M}_2$ — is not empty and, hence, the components can indeed operate together. This is because, by Theo. 16, $\Lambda_{\mathcal{M}_1 \| \mathcal{M}_2} \precapprox \Lambda$ and, hence, if $\mathcal{M}_1 \parallel \mathcal{M}_2$ is consistent, $\Lambda_{\mathcal{M}_1}^{\iota_1} \cap \Lambda_{\mathcal{M}_2}^{\iota_2}$ is not empty.

4.3 Feasibility

The property of being DP-enabled is related to a stronger notion of consistency called 'feasibility': whereas consistency guarantees the existence an execution, feasibility requires that, no matter what inputs the machine receives from its environment, it can produce an execution.

Definition 29 (Feasible) *A DTIOM $\mathcal{M}$ is said to be* feasible in relation to *$J \subseteq Act_{\mathcal{M}}^I$ and a multiple δ of $\delta_{\mathcal{M}}$ if, for every δ-timed trace λ over J and state (l, v) reachable at a time T such that $(T + \delta_{\mathcal{M}})$ is a multiple of δ, there is an execution starting at (l, v) that generates a $\delta_{\mathcal{M}}$-timed trace λ' such that $\lambda'|_J \preceq \lambda$, where $\lambda'|_J$ is the timed trace obtained from λ' by forgetting the elements in $Act_{\mathcal{M}} \setminus J$ from the underlying action sequence. A DTIOM $\mathcal{M}$ is said to be* feasible *if it is feasible in relation to $Act_{\mathcal{M}}^I$ and $\delta_{\mathcal{M}}$.*

This notion of feasibility is similar to the one use, for example, in [14], which we have relativised to given sets of input actions in order to account for structured interactions with the environment.

Proposition 30 *A DTIOM $\mathcal{M}$ that makes independent progress and is DP-enabled in relation to $J \subseteq Act^I_{\mathcal{M}}$ and a multiple δ of $\delta_{\mathcal{M}}$ is feasible in relation to J and δ.*

In relation to the compositionality of feasibility, we can prove:

Theorem 31 *Let $\mathcal{M}_i$, $i = 1, 2$, be two δ-compatible DTIOMs that can make independent progress. Let δ' be a common multiple of δ_1 and δ_2 and δ'_1 a multiple of δ_1 and $J \subseteq Act^I_1$. If (a) $\mathcal{M}_1$ is DP-enabled in relation to J and δ'_1, (b) $\mathcal{M}_1$ is DP-enabled in relation to $Act^I_1 \cap Act^O_2$ and δ', (c) $\mathcal{M}_2$ is DP-enabled in relation to $Act^I_2 \cap Act^O_1$ and δ', and (d) both $\mathcal{M}_1$ and $\mathcal{M}_2$ are δ'-cooperative in relation to $Act_1 \cap Act_2$, then $\mathcal{M}_1 \parallel_\delta \mathcal{M}_2$ is feasible in relation to $J \setminus Act^O_2$ and δ'_1.*

5 Related Work

Several researchers have recently addressed discrete timed systems with heterogeneous clock granularities. However, the main focus has been either on specification or on modelling and simulation, not so much on the challenges that heterogeneity raises on run-time interconnection of systems. For example, Forget et al. propose in [10] a synchronous data-flow language that supports the modelling of multi-periodic systems. In this setting, each system has its own discrete periodic clock granularity; composition is supported by a formal clock calculus that allows, in particular, for the refinement of clock granularities in a way that is similar to what we propose in Sec. 3. Aside from the fact that we adopt an automata-based representation, the main difference with our work is that they leave open the question of component-based verification of properties such as consistency.

Similarly, in [6], the authors introduce a formal communication model of behaviour for the composition of heterogeneous real-time cyber-physical systems based on logical clock constraints. Although this model supports the combination of heterogeneous timed systems, the authors do not consider the particular case of discrete periodic systems. In [17], the authors present a methodology (ForSyDe) for high-level modelling and refinement of heterogeneous embedded systems; whilst the semantics they propose, and the notion of clock-refinement they introduce, are similar in essence to ours, their main focus is again on modelling and simulation, whereas ours is on the structures that support compositional reasoning over properties of interconnected systems.

To cope with heterogenous time scales, several approaches to the specification of real-time systems, notably the Timebands Framework [5], have also adopted an explicit representation of time granularity. That framework, unlike others, does not require that all descriptions be transformed into the finest granularity.

Some attempts have also been made at addressing compositionality, for example in [15] that exploits the concept of tag machines [2]. However, the notion of composition of systems introduced by the authors (using tag morphisms) is more relaxed than ours in that it allows for the delay between events to be modified in given tag machines. A consequence of this generality is that the language

resulting from a composition is not an approximation of the intersection of the original languages, which, as argued in our paper, is essential for addressing global properties of interconnected systems as implemented.

From a practical point of view, some tools have been developed for modelling and simulating heterogeneous systems. For example, Ptolemy Classic [4] introduced the concept of heterogeneous combinations of semantics such as asynchronous models with timed discrete-events models. The concept was picked up in other tools such as System C [12], Metropolis [11] and Ptolemy II [16]. The common characteristics of these tools is that (1) they are based on a model that is more general than the one we propose in this paper, and (2) they do not consider composition of discrete timed systems with different periodic clocks. As a consequence, they are not able to provide results as strong as ours when it comes to reasoning about specific global properties of interconnected systems.

6 Concluding remarks

This paper proposes a new theoretical framework for the compositional design of timed heterogeneous systems based on an extension of timed input/output automata [14, 7] where automata are assigned a clock granularity (what we call timed machines). Composition is thus extended to cater for automata that operate over different clock granularities.

One key aspect of our work is that we support the design of heterogeneous timed systems whose clock granularities can be made compatible without modifying the time domains of the individual components. This is important so that components can be interconnected at run time, not design time, which is essential for addressing the new generation of systems that are operating in cyberspace, where they need to be interconnected, on the fly, to other systems. Our approach is truly compositional in that we can obtain properties of a whole system of interconnected components without having to compute their composition.

The main properties that we address are consistency (there exists at least a joint trace on which all components can agree) and feasibility (there exists at least a joint trace on which all components can agree no matter what input they receive from their environment). The technical results that support compositional verification of consistency and feasibility are based on new notions of time refinement and of cooperation conditions through which timed components can be ensured to be open to interactions with other components across different time granularities.

There are two main directions for future work. The first is to implement and evaluate our approach on concrete case studies. A possibility would be to implement the framework as an extension of Ptolemy [4], which would give us access to industrial-size case studies. The second aims at extending our work to networks of heterogeneous timed systems that communicate asynchronously by building on [8] and [9].

Acknowledgments

This work was partially supported by the Royal Society International Exchange grant IE130154.

References

1. M. Abadi and L. Lamport. The existence of refinement mappings. *Theor. Comput. Sci.*, 82(2):253–284, 1991.
2. A. Benveniste, B. Caillaud, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. Tag machines. In *EMSOFT*, pages 255–263. ACM, 2005.
3. M. Broy and K. Stølen. *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.
4. J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogenous systems. *Int. Journal in Computer Simulation*, 4(2):155–182, 1994.
5. A. Burns and I. J. Hayes. A timeband framework for modelling real-time systems. *Real-Time Syst.*, 45(1-2):106–142, June 2010.
6. Y. Chen, Y. Chen, and E. Madelaine. Timed-pNets: A formal communication behavior model for real-time CPS system. In *Trustworthy Cyber-Physical Systems*. School of Computing Science Technical Report Series, Newcastle University, 2012.
7. A. David, K. G. Larsen, A. Legay, U. Nyman, and A. Wasowski. Timed I/O automata: a complete specification theory for real-time systems. In *HSCC*, pages 91–100. ACM, 2010.
8. B. Delahaye, J. L. Fiadeiro, A. Legay, and A. Lopes. A timed component algebra for services. In D. Beyer and M. Boreale, editors, *FORTE*, volume 7892 of *LNCS*, pages 242–257. Springer, 2013.
9. J. L. Fiadeiro and A. Lopes. An interface theory for service-oriented design. *Theor. Comput. Sci.*, 503:1–30, 2013.
10. J. Forget, F. Boniol, D. Lesens, and C. Pagetti. A multi-periodic synchronous data-flow language. In *HASE*, pages 251–260. IEEE Computer Society, 2008.
11. G. Gößler and A. L. Sangiovanni-Vincentelli. Compositional modeling in metropolis. In *EMSOFT*, volume 2491 of *LNCS*, pages 93–107. Springer, 2002.
12. T. Grötker. *System Design with SystemC*. Springer, 2002.
13. T. A. Henzinger, Z. Manna, and A. Pnueli. Timed transition systems. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *REX Workshop*, volume 600 of *LNCS*, pages 226–251. Springer, 1991.
14. D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *The Theory of Timed I/O Automata*. Morgan & Claypool Publishers, 2006.
15. T. Le, R. Passerone, U. Fahrenberg, and A. Legay. A tag contract framework for heterogeneous systems. In C. Canal and M. Villari, editors, *ESOCC Workshops*, volume 393 of *CCIS*, pages 204–217. Springer, 2013.
16. E. A. Lee and H. Zheng. Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems. In *EMSOFT*, pages 114–123. ACM, 2007.
17. I. Sander and A. Jantsch. System modeling and transformational design refinement in ForSyDe [formal system design]. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 23(1):17–32, 2004.